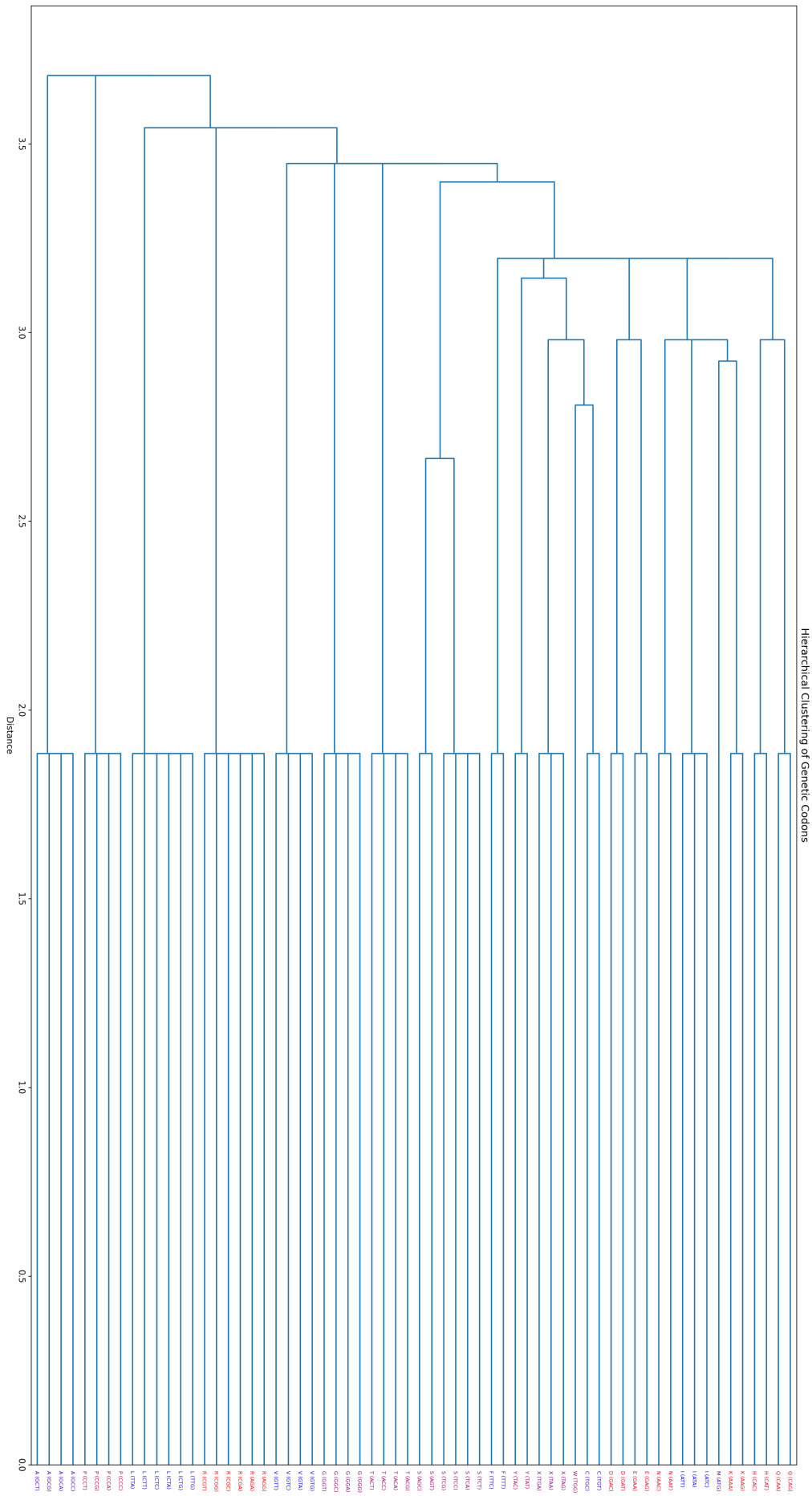
Codon Cluster Analysis With Hydropathy Written by GPT-4 in Python

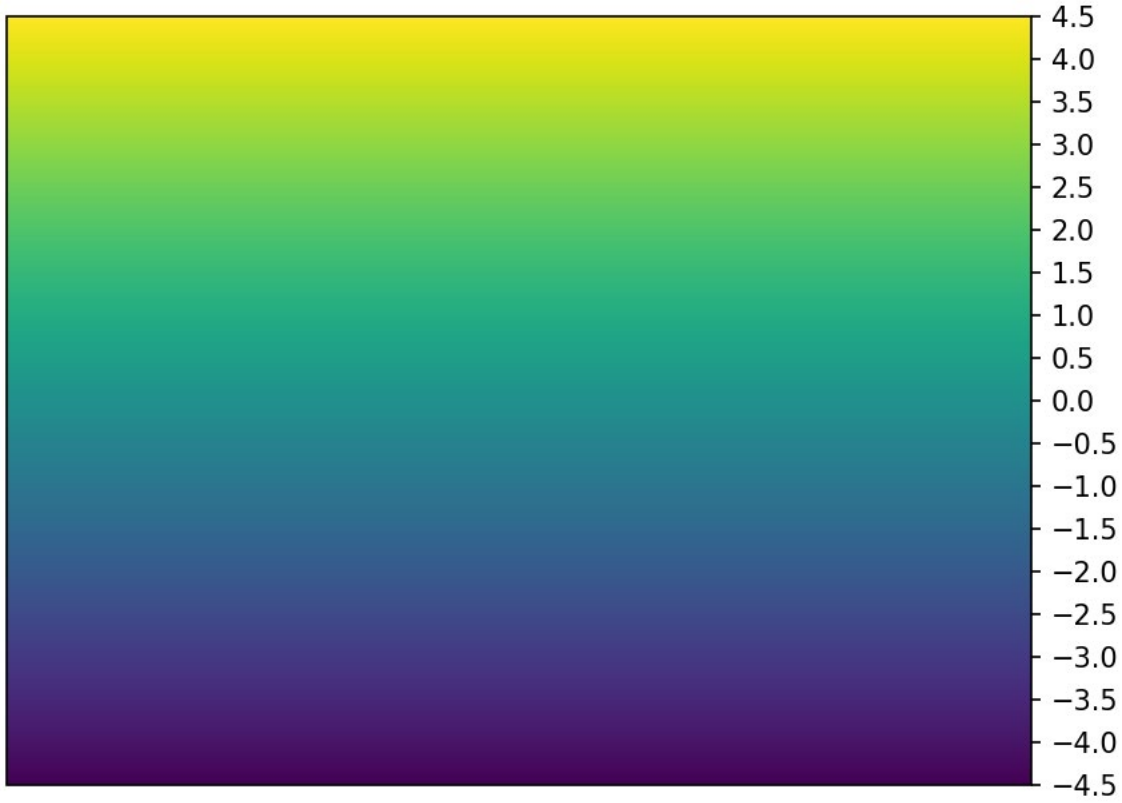
Douglas C. Youvan

doug@youvan.com

October 25, 2023

The Python code is included in this paper.


Hierarchical Clustering of Genetic Codons
Distance
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
A (GCT)
A (GCG)
A (GCA)
A (GCC)
P (CCT)
P (CCG)
P (CCA)
P (CCC)
L (CTT)
L (CTC)
L (CTA)
L (CTG)
L (TTG)
R (CGT)
R (CGC)
R (CGA)
R (AGA)
R (AGG)
V (GTT)
V (GTC)
V (GTA)
V (GTG)
G (GGT)
G (GGC)
G (GGA)
G (GGG)
T (ACT)
T (ACC)
T (ACA)
T (ACG)
S (AGC)
S (AGT)
S (TCG)
S (TCA)
S (TCT)
S (TCC)
F (TTC)
F (TTT)
Y (TAC)
Y (TAT)
X (TAA)
X (TAA)
X (TAG)
W (TGG)
C (TGC)
C (TGT)
D (GAC)
D (GAT)
E (GAA)
E (GAG)
N (AAC)
N (AAT)
I (ATT)
I (ATC)
M (ATG)
K (AAA)
K (AAG)
H (CAC)
H (CAT)
Q (CAA)
Q (CAG)



Kyte-Doolittle Hydropathy Scale

```
import numpy as np
from scipy.spatial.distance import hamming, pdist
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
import matplotlib as mpl

# Use the "viridis" colormap
cmap = plt.get_cmap('viridis')

# Create mappable object for the colormap
norm = mpl.colors.Normalize(vmin=-4.5, vmax=4.5)
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])



# Define a custom colormap
colors = [(1, 0, 0), (0, 0, 1)]  # deep red to deep blue
cm_name = 'custom_div_cmap'
cm = mpl.colors.LinearSegmentedColormap.from_list(cm_name,
colors, N=100)

# Create mappable object for the new colormap
norm = mpl.colors.Normalize(vmin=-4.5, vmax=4.5)
```

```
sm = plt.cm.ScalarMappable(cmap=cm, norm=norm)
sm.set_array([])



# Step 1: Representing the Genetic Code
genetic_code = {
    'ATA': 'I', 'ATC': 'I', 'ATT': 'I', 'ATG': 'M',
    'ACA': 'T', 'ACC': 'T', 'ACG': 'T', 'ACT': 'T',
    'AAC': 'N', 'AAT': 'N', 'AAA': 'K', 'AAG': 'K',
    'AGC': 'S', 'AGT': 'S', 'AGA': 'R', 'AGG': 'R',
    'CTA': 'L', 'CTC': 'L', 'CTG': 'L', 'CTT': 'L',
    'CCA': 'P', 'CCC': 'P', 'CCG': 'P', 'CCT': 'P',
    'CAC': 'H', 'CAT': 'H', 'CAA': 'Q', 'CAG': 'Q',
    'CGA': 'R', 'CGC': 'R', 'CGG': 'R', 'CGT': 'R',
    'GTA': 'V', 'GTC': 'V', 'GTG': 'V', 'GTT': 'V',
    'GCA': 'A', 'GCC': 'A', 'GCG': 'A', 'GCT': 'A',
    'GAC': 'D', 'GAT': 'D', 'GAA': 'E', 'GAG': 'E',
    'GGA': 'G', 'GGC': 'G', 'GGG': 'G', 'GGT': 'G',
    'TCA': 'S', 'TCC': 'S', 'TCG': 'S', 'TCT': 'S',
    'TTC': 'F', 'TTT': 'F', 'TTA': 'L', 'TTG': 'L',
    'TAC': 'Y', 'TAT': 'Y', 'TAA': '_', 'TAG': '_',
    'TGC': 'C', 'TGT': 'C', 'TGA': '_', 'TGG': 'W',
}
```

```
# Replace _ with X for stop codons
for codon, aa in genetic_code.items():
    if aa == '_':
        genetic_code[codon] = 'X'

# Step 2: Calculate Pairwise Distances
def nucleotide_distance(codon1, codon2):
    return hamming(list(codon1), list(codon2))

def functional_distance(codon1, codon2):
    return 0 if genetic_code[codon1] == genetic_code[codon2] else
1

codons = list(genetic_code.keys())
distance_matrix = np.zeros((64, 64))
for i in range(64):
    for j in range(64):
        distance_matrix[i][j] = nucleotide_distance(codons[i],
codons[j]) + functional_distance(codons[i], codons[j])

# Convert square distance matrix to condensed form
condensed_distance_matrix = pdist(distance_matrix)
```

```
# Step 3: Hierarchical Clustering
linked = linkage(condensed_distance_matrix, method='single')

# Create a new list of labels with single-letter codes
new_labels = [genetic_code[codon] + ' (' + codon + ')' for codon in codons]

# Plot the dendrogram
fig, ax = plt.subplots(figsize=(10, 14))
dendrogram(linked, orientation="left", labels=new_labels, color_threshold=0)

# ... [Previous code remains unchanged]

# Define the Kyte-Doolittle hydropathy values and coloring function
hydropathy_values = {
    'A': 1.8, 'R': -4.5, 'N': -3.5, 'D': -3.5, 'C': 2.5,
    'Q': -3.5, 'E': -3.5, 'G': -0.4, 'H': -3.2, 'I': 4.5,
    'L': 3.8, 'K': -3.9, 'M': 1.9, 'F': 2.8, 'P': -1.6,
    'S': -0.8, 'T': -0.7, 'W': -0.9, 'Y': -1.3, 'V': 4.2,
    'X': 0  # for stop codons, neutral value
}
```

```
# Define a custom colormap
colors = [(1, 0, 0), (0, 0, 1)]  # deep red to deep blue
cm_name = 'custom_div_cmap'
cm = mpl.colors.LinearSegmentedColormap.from_list(cm_name, colors, N=100)

# Create mappable object for the new colormap
norm = mpl.colors.Normalize(vmin=-4.5, vmax=4.5)
sm = plt.cm.ScalarMappable(cmap=cm, norm=norm)
sm.set_array([])


# Adjust the color mapping
for label in ax.get_ymajorticklabels():
    label_text = label.get_text().split()[0]  # Extract the amino acid
    if label_text in hydropathy_values:
        value = hydropathy_values[label_text]
        color = sm.to_rgba(value)
        label.set_color(color)


plt.title('Hierarchical Clustering of Genetic Codons')
plt.xlabel('Distance')
```

```
plt.tight_layout()
plt.show()
```

```
import matplotlib.pyplot as plt
import numpy as np

# Define the Kyte-Doolittle scale values range
norm = plt.Normalize(-4.5, 4.5)

# Use the viridis colormap
cmap = plt.get_cmap('viridis')

sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])

# Create a new axis for the colorbar
cax = plt.axes([0.1, 0.1, 0.8, 0.8])
cbar = plt.colorbar(sm, cax=cax, orientation='vertical')
```

```
cbar.set_label('Kyte-Doolittle Hydropathy Value', rotation=270, labelpad=20)

cbar.set_ticks(np.linspace(-4.5, 4.5, 19))  # Set ticks from -4.5 to 4.5 with an interval of 0.5

cbar.ax.tick_params(labelsize=10)


plt.show()
```